Développement Très Rapide en Applications de Gestion

Créer des logiciels de gestion rapidement

- Matthieu GIROUX - www.liberlog.fr.

- Développeur Indépendant.
- Installation et personnalisation Web.
- Edition de livres : LAZARUS et Ecriture.
- Création de Logiciels de Gestion.
- Création d'un savoir-faire.

Table des matières

1.1) Présentation : Pourquoi rapide ?

Actuellement l'informatique permet de gagner du temps.

Seulement la création d'un logiciel est de plus en plus longue. Beaucoup de projets logiciels sont arrêtés.

Or l'informatique sert à automatiser.

Il est possible d'automatiser la création logicielle.

1.2) Présentation : L'utilisateur ?

L'utilisateur se pose toujours les mêmes questions.

L'ergonomie répond donc toujours aux mêmes questions.

Il est possible de préparer l'interface avec l'utilisateur, avant même que le logiciel soit créé, comme le font certains logiciels.

Comment ?

Avec les interfaces conçues précédemment, et un travail à effectuer automatisé.

1.3) Présentation : L'interface

Créer chaque interface entièrement :

- Crée des problèmes d'ergonomie prévisibles
- Fait perdre du temps : A créer et à corriger
- Fait perdre en fiabilité : Beaucoup pour rien

C'est connu : Beaucoup de logiciels deviennent lourds et mal faits, parce qu'il y a du copié-collé, parce qu'on mélange la partie technique avec ce que demande l'utilisateur, le métier.

1.4.1) Présentation : Gagner en temps et en fiabilité

En centralisant chaque élément du logiciel :

- On anticipe l'ergonomie du logiciel
- On fiabilise l'outil : Les tests font l'essentiel
- On participe aux projets qu'on utilise
- On crée son savoir-faire et sa valeur ajoutée

En centralisant on finit par créer un savoir-faire de développement très rapide de logiciels.

1.4.2) Les composants RAD : Gagner en temps et en fiabilité

Les composants RAD Lazarus sont installables en un simple clic de souris.

L'inspecteur d'objets permet alors de gagner jusqu'à trois fois plus de temps si on crée ses composants.

Le composant va alorrs servir à créer le moteur VRAD, lisible à partir de simples fichiers passifs au format Business Process Management.

2.1) Définition : Rapid Application Development

Rapid Application Development ou RAD

=

Développement Rapide d'Applications ou DRA

= Créer visuellement pour créer vite

Le Développement Très Rapide d'Applications (DTRA ou VRAD) va être présenté.

Le Développement Très Rapide d'Applications permet de créer votre logiciel de gestion d'entreprise personnalisé.

2.2) Mauvais exemples
Les API de développement

Une API est une bibliothèque à programmer

Les API de gestion nécessitent :

- De créer des sources difficiles à manipuler
- De programmer pour réinventer la roue

Un ingénieur développeur peut transformer des API de gestion afin de créer l'interface du logiciel de gestion à partir de certains fichiers de librairies VRAD.

2.3) Comment bien créer une librairie ?

Chronologie de création d'un savoir-faire

- Au début on crée des unités de fonctions.
- Puis on utilise et surcharge des composants.
- On crée des paquets de composants.
- On ouvre alors sa librairie aux autres API.
- On automatise les paquets en une librairie.
- La librairie nécessite peu de code ou aucun.

2.4) Histoire : Rapid Application Development

Un logiciel est composé de :

- Une partie métier : Ce que veut le client.
- Une partie technique : L'informatique.

Que ce soit avec sans des outils RAD on :

- Mélangeait la technique et le métier.
- Refaisait le logiciel entièrement.

La partie métier du logiciel doit être gardée.

2.5) Histoire : Rapid Application Development

On s'aperçoit que le client final doit savoir comment fonctionne un logiciel mais il n'aime pas parler technique.

Il est possible d'éluder la partie technique en créant le logiciel à partir de la demande de l'utilisateur.

Le client doit cependant savoir :

- Comment on fait son logiciel (RAD, VRAD, etc).
- S'il a la main sur le logiciel créé.

2.6) Histoire : Rapid Application Development

Des Outils de Développement Rapide sont au départ créés pour fidéliser leurs clients :

- VISUAL BASIC, DELPHI, LAZARUS.

Maintenant des savoirs-faire peuvent s'y ajouter. Ils permettent de garder les informations importantes :

- GLADE GTK pour des interfaces simples.
- JAVA LEONARDI et WAVE MAKER pour la gestion.
- GAME MAKER pour les jeux.

3.1) Les logiciels de gestion

Un logiciel de gestion c'est :

- Un logiciel d'entreprise
- L'administration d'un site web
- Une comptabilité d'entreprise

On s'aperçoit qu'il est facile de modéliser un logiciel d'entreprise, car un logiciel de gestion répète les mêmes procédés.

3.2) Les logiciels de gestion

Un logiciel de gestion c'est :

- Une liaison vers un serveur de données.
- Des relations entre les données.
- Des statistiques, des tableaux de calcul.
- De la cartographie, d'autres ajouts.

Tout ceci n'est-il pas défini donc automatisable ?

Un logiciel VRAD automatise ces procédés.

3.3) Le Multi-plateforme

Le prestataire veut utiliser son propre savoir-faire.

Le multi-plateforme c'est :

- Etre indépendant de l'environnement.
- Etre indépendant du savoir-faire utilisé ?

Il est possible déjà de changer de savoir-faire informatique grâce aux données.

Il est possible d'être indépendant de tout savoir-faire utilisé grâce au VRAD.

3.4) Développement Rapide d'Application (DRA ou RAD)

- Créer visuellement une application,
- Pour gagner du temps dans la création,
- Afin de créer une application intuitive.

La plupart des outils RAD n'automatisent pas assez la gestion d'une entreprise.

Le Very Rapid Application Development est l'amélioration du RAD pour les serveurs de gestion ou d'autres interfaces avec des répétitions.

3.5) Créer son interface avec des fichiers : le VRAD

Il est déjà possible de créer une interface de gestion grâce à l'Ingénierie Pilotée par Modèles.

Un fichier passif contenant la partie métier est lu et crée l'interface grâce au moteur VRAD, en plus du savoir-faire RAD.

- GLADE GTK permet de créer une interface standard, seules les actions sont codées.
- LEONARDI et WAVE MAKER permettent de créer une interface de gestion.
- LIBERLOG possède un moteur VRAD, basé sur LEONARDI.

3.6) Créer son interface avec des fichiers : le VRAD

Les fichiers passifs :

- Permettent de créer des modèles d'objets.
- Ne contiennent que les demandes du client.
- Nécessitent d'automatiser la présentation.

Il est possible de créer des thèmes de présentations. Ainsi l'interface change en fonction de ce que souhaite l'utilisateur.

4.1) Intérêts du Développement Très Rapide d'Application

Le Développement Très Rapide permet :

- D'empêcher mieux les erreurs de se produire.
- De ne créer au final que l'analyse du logiciel.
- De gagner du temps dans la création.
- D'être indépendant de tout savoir-faire.
- Que le programmeur pense fonctionnalités.

Le code créé sera réutilisable, centralisé, facilement utilisable, intégré plus facilement.

4.2) Développement Rapide vs Ligne de commande

Exemple : Création d'une fiche HTML simple

Un code centralisé utilisé avec du copié-collé.

- 3 jours et ça n'est peut-être pas fini.

La même chose avec un outil RAD

- ½ journée d'analyse et ½ journée de création.
- Le composant automatise certaines créations.
- La fiche est utilisable sans avoir trop à tester;

4.3) Fichiers passifs vs RAD classique

Les fichiers passifs :

- Permettent de modéliser le coeur de métier
- Peuvent être créés à partir d'une analyse
- Rendent indépendants du savoir-faire utilisé
- Sont définis et peuvent évoluer
- Permettent de créer d'autres interfaces
- Permettent de penser fonctionnalités

C'est l'analyse d'½ journée qui crée le logiciel.

L'analyse correspond au logiciel créé.

4.4) Fichiers passifs vs RAD classique

Avec les fichiers passifs on :

- Réfléchit fonctionnalités et coeur de métier.
- Détermine ce qui est faisable rapidement.
- Détermine ce qui n'est pas modélisable.
- Crée des plugins pour ce qui n'est pas fait.
- Sait où l'on va.

Une fois le logiciel créé on peut créer d'autres genres d'interfaces avec des savoirs-faire VRAD.

4.4) Qualité VRAD

Avec un moteur VRAD :

- On gagne du temps et est plus agile.
- On facilite la mise en place de futurs logiciels.
- Ne teste que le moteur, pas l'interface créée.
- L'analyse modélisée crée le logiciel.
- La maintenance est centralisée.
- Le développeur va à l'essentiel.

5.1) Création de plugins VRAD

La création d'un plugin VRAD :

- Répond à une micro ou une macro demande
- Sera intégrée dans les fichiers passifs.
- Se fera rapidement si on utilise un EDI RAD.
- Sera acquise une fois le plugin créé.
- Nécessitera de créer le plugin sur d'autres EDI.
- Sera modélisable dans l'analyse.

6.1) VRAD LEONARDI GPL

Cette librairie permet de :

- Créer des fichiers passifs avec l'analyse.
- Créer le logiciel avec les fichiers passifs.
- Créer la partie technique en plugins.
- Faire du Reverse Engeenering de données.
- Créer un logiciel à la fois WEB et non WEB.

6.2) VRAD LEONARDI GPL

Avec LEONARDI on peut dans les IHM :

- Gérer avec des formulaires.
- Trier, filtrer, rechercher, composer.
- Imprimer, exporter, importer.
- Créer des statistiques, arbres, tableaux.
- Créer des diagrammes, des cartes.
- Créer des plugins liés aux fichiers passifs.
- Apprendre facilement grâce aux docs.

6.3) VRAD LEONARDI GPL

LEONARDI permet de réaliser votre :

- Gestion de Chaîne Logistique (GCL ou SCM).
- Gestion de Relation Client (GRC ou CRM).
- Supervision, Administration de réseau...
- Configuration : d'équipements réseaux…
- Système Information Communication (SIC).
- Système d'Aide au Commandement.
- Progiciel de Gestion Intégré (PGI ou ERP).
- Gestion de Référentiels.
- Système d'Information Géographique (SIG).
- Gestion de stocks.

Il est possible de réaliser un prototypage rapide personnalisé.

6.4) LEONARDI – RESTRICTION

La librairie LEONARDI est gratuite :

- Pour créer des logiciels commerciaux.
- En utilisant les SGBD gratuits.

- Les liens de données sont payants vers les SGDB payants.
- La cartographie et le diagramme de GANTT sont payants.

6.5) LEONARDI – POSSIBILITES

Améliorations de LEONARDI :

- Création de plugins et de composants libres.
- Compatibilité théorique avec JELIX JFORMS.
- Transfert vers d'autres outils avec les fichiers passifs.

6.6) LEONARDI et JELIX JFORMS Frameworks complémentaires

L'analyse LEONARDI permet de :

- Créer un logiciel.
- En utilisant des fichiers passifs de formulaires.
- Qui créent le logiciel.

Il est possible en théorie de traduire certains fichiers entre LEONARDI et JELIX JFORMS.

On est alors indépendant de tout outil.

7.1) JELIX – LGPL

Cette librairie permet de :

- Créer un logiciel de gestion avec le plugin JFORMS.
- Créer la partie technique en plugins JELIX.
- Créer un logiciel ou un site WEB.

Avantages

- Beaucoup de plugins pour son portail WEB.
- Possibilité de convertir des fichiers XML.

Il est possible de transférer l'analyse de LEONARDI.

7.1) WAVE MAKER – Licence Apache

Cette librairie permet de :

- Créer un logiciel de gestion facilement.
- D'utiliser SPRING.
- Créer un logiciel ou un site WEB.

Avantages

- Si on est technicien ou analyste on peut créer un logiciel.

7.1) GAME MAKER – Licence Commerciale

Cette librairie permet de créer un jeu facilement.

Avantages

- Le jeu est vite fait.
- On n'est pas obligé de connaître la programmation.

Inconvénients

- Si le jeu se complique on doit payer une licence d'utilisation.
- On devient dépendant du savoir-faire utilisé

8.1) Pourquoi utiliser un EDI RAD ?

- Evolutions rapides.
- Les composants sont vite mis en place.
- La structure des composants est homogène.
- Maintenance facile.
- Centralisation et individualisation des sources.
- Pas de création inutile.
- Séparation selon les parties techniques.

8.2) LAZARUS

Avantages

- Projet libre, réutilisable à la vente et participatif.
- Sur WINDOWS LINUX UNIX MAC-OS BSD.
- Beaucoup de composants DELPHI libres.
- Exécution rapide car non retraduite.
- Un exécutable indépendant par plateforme.
- Création rapide si maitrisée.

8.3) LAZARUS

Inconvénients

- Poids des exécutables important.
- Début:1999 (compatible DELPHI à 90%).
- Nécessite de réécrire la partie WINDOWS.
- Partie graphique DELPHI compatible à 99 %.
- Composants traduits ont moins de propriétés.
- Utiliser les unités multi-plateformes.
- Plus complet sous WINDOWS, puis LINUX.

8.5) Comment bien créer un composant RAD ?

Comment bien travailler ?

- Utilisation facile du composant.
- Evolutivité.
- Portabilité.
- Interopérabilité avec les autres composants.
- Anticipation sur la structure du composant.
- Méthodes et variables en anglais adéquate.

8.6) Le potentiel LAZARUS

LAZARUS est un EDI RAD qui dispose :

- Du framework de LIBERLOG.FR.
- De la gestion des données.
- D'Exécutables visuels WINDOWS,LINUX,MAC.
- De l'embarqué sur certains téléphones mobiles.
- D'une création WEB ou pas par composants.

9.1) FRAMEWORK LIBERLOG

XML FRAMES

- Créer des logiciels de gestion.
- Grâce aux composants RAD de gestion.
- Créant vite des fiches simples.
- Réutilisation de LEONARDI.
- Possibilités de compatibilté JELIX JFORMS.

Il sera avec possible de créer des logiciels embarqués.

9.2) Pourquoi un savoir-faire ?

Le savoir-faire utilisé :

- C'est ce qui permet de créer les interfaces.
- Permet d'être indépendant du prestataire s'il est libre et si on demande les sources du logiciel.
- Peut centraliser la partie métier si on le demande.

Si la partie métier n'est pas centralisée alors on remarque un décalage entre l'analyse et la création du logiciel de gestion.

9.3) Créer un savoir-faire VRAD

- MICROSOFT possède une organisation qui n'est pas favorable à l'indépendance de ses clients pour son futur outil VRAD.
- Seuls les PME ou clients finaux amélioreront un savoir-faire libre en VRAD de gestion.
- Il faut utiliser les sources libres à disposition et créer un format de fichiers VRAD unique.
- La création d'un savoir-faire libre en VRAD permet de récupérer la partie métier du logiciel.